VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Hra pro mobilní telefon

Mobile phone game

2009 **Tomáš Válek**

zde bude umístěno zadání bakalářské práce, což je oficiální druhý list práce

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace ze kterých jsem čerpal.

V Ostravě dne: Podpis: ……..

Abstrakt:

Náplní této bakalářské práce byla implementace hry Lodě pro mobilní telefon na platformě Java ME. Přesněji se jednalo o implementaci hry pro jednoho hráče proti počítači, dále hry dvou hráčů na jednom zařízení a v poslední fázi byla realizována implementace hry mezi dvěma zařízeními, kdy data jsou přenášena pomocí bezdrátové technologie známé pod obchodním názvem Bluetooth. Samozřejmostí byl vlastní návrh grafického zpracování samotné hry a jejího GUI, stejně tak navržení a implementace hry proti počítači a hry přes Bluetooth.

Klíčová slova:

Java ME, Bluetooth, JSR 82, GameCanvas, Sprite, MIDlet, Layer, CLDC, CDC, MIDP, NetBeans

Abstract:

The goal of this bachelor thesis was an implementation of mobile phone game Ship on Java ME platform. More precisely it contains an implementation of single player game against computer, next is two players game on one device and finally multiplayer game with support of commercial well known Bluetooth wireless technology. Obvious I designed my own graphic of the game and its GUI, and of course I implementated player versus artifical inteligence game and Bluetooth game.

Keywords:

Java ME, Bluetooth, JSR 82, GameCanvas, Sprite, MIDlet, Layer, CLDC, CDC, MIDP, NetBeans

Seznam použitých symbolů a zkratek:

API – Application Programming Interface
CDC – The Connected Device Configuration
CLDC – The Connected Limited Device Configuration
GUI – Graphical User Interface
IDE – Integrated Development Environment
ISM – Industrial-Scientific-Medical
Java ME – Java Platform, Micro Edition
JVM – Java Virtual Machine
MIDP – Mobile Information Device Profile
PHP – Hypertext Preprocesor, původně Personal Home Page
SMS – Short message service
UUID – Universally Unique Identifier
VM – Virtual Machine
VMD – Visual Mobile Designer

Obsah

1. Úvod

V dnešní době je svět takříkajíc zaplaven mobilními telefony, ve vyspělých zemích se počty telefonů pohybují více jak 1 mobil připadající na jednoho obyvatele. U nás se mobily začaly pomalu prosazovat od začátku 90. let, kdy první přístroje uměly jen telefonovat, vážily i více než jeden kilogram a napájecí baterie nevydržela ani jeden den. Jejich cena se pohybovala kolem 50 000 korun. Od té doby se mnoho změnilo, dnes máme mobily lehké, baterie vydrží i několik dní, máme velké barevné displeje, vestavěné fotoaparáty, rádia a další věci.

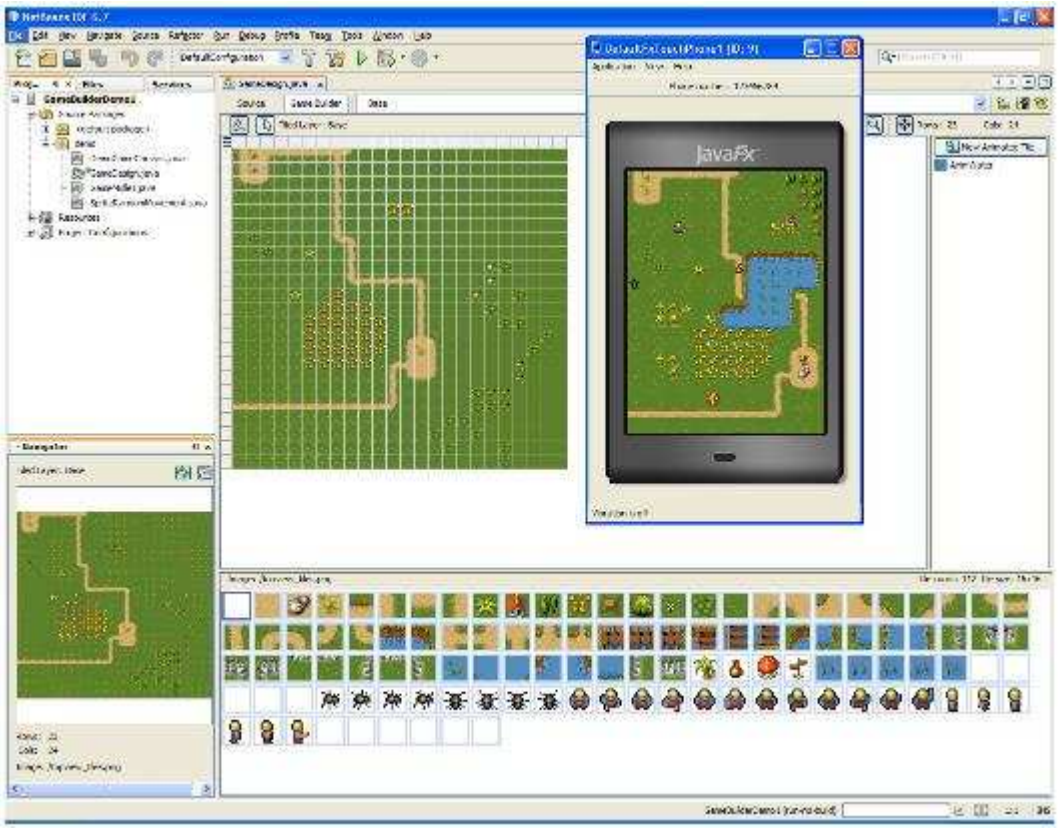
Hry na mobilních telefonech se objevily, jakmile mobily přestaly mít displej na který se vešel jen jeden řádek textu. Asi nejznámější hrou konce 90. let byl Had od firmy Nokia, který dokonce podporoval hru dvou hráčů přes infraport. V té době se hrály většinou hry, které byly do přístroje nainstalovány už od výrobce, až později se rozběhl obchod s Java hrami, který dneska dosáhl slušných rozměrů, je to způsobeno jednoduchostí objednání hry i tím, že ceny se pohybují zhruba od 30 do 80 korun, což není zase tak moc na to, že některé hry poskytují plno hodin zábavy. Už před pár lety mě lákala chuť něco pro mobilní telefony vytvořit, ale nikdy jsem se k tomu nedostal. Až tato práce mi to umožnila a jsem rád, že jsem poznal jaké to je vyvíjet aplikaci pro mobilní telefon a hlavně jsem poznal, že vyvíjet hru není vůbec lehká záležitost, protože některé věci, které se mi na začátku jevily jako snadné se později ukázaly být celkem složitými.

Text jsem rozdělil do několika kapitol, první je tento úvod, dále se něco dozvíte o vývojovém prostředí ve kterém jsem pracoval, o použitých technologiích a nakonec o samotné implementaci, kde popíšu co a jak se mi podařilo udělat a také zde okomentuji případné nedostatky, které by se daly do budoucna vylepšit.

2. Vývoj hry

2.1. Vývojové prostředí NetBeans 6.5

Tento projekt jsem programoval ve vývojovém prostředí NetBeans 6.5, které mi poskytlo vše co jsem ke zdárnému dokončení projektu potřeboval. V současné době je již k dispozici verze 6.7. Jedná se o tzv. „free open source“ neboli volně šiřitelný software s otevřeným zdrojovým kódem. Je v něm zahrnuta podpora pro vývoj desktopových, enterprise, webových a mobilních aplikací s podporou programovacích jazyků Java, C/C++ a dokonce i v dynamických jazycích jako je PHP, JavaScript, Groovy a Ruby. NetBeans se lehce instaluje a snadno používá, na internetu lze nalézt velké množství tutoriálů, které usnadní práci s tímto IDE. Podporuje platformy jako Windows, Linux, Mac OS X a Solaris. Čerpáno z [1].



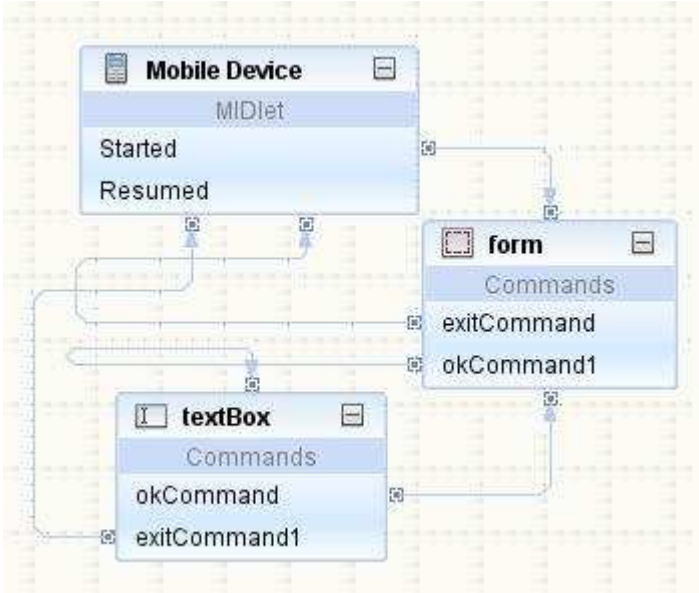
Obr. č.1 : NetBeans a GameBuilder

Na přiloženém obrázku č.1 je zachycena obrazovka NetBeans IDE se spuštěným integrovaným nástrojem pro tvorbu her s názvem Mobile Game Builder, kde se dají jednoduše vytvářet různé scenérie, vytvářet animace spritů, umisťovat sprity a další věci.

Na začátku jsem v Game Builderu zkoušel tvořit, ale dost mi tam vadila ta vlastnost, že některé části kódu nešly vůbec editovat, což bylo velmi svazující a brzo jsem tento způsob tvorby opustil a vydal jsem se cestou tvorby vlastního kódu podle svých představ.

NetBeans IDE umožňuje vývoj aplikací pro mobilní telefony, PDA, Set-top boxy a další. Umožňuje vytvářet, testovat a debugovat aplikace pro MIDP 1.0, 2.0, 2.1, CLDC 1.0 a 1.1 a CDC. K NetBeans je přibalována poslední verze Java ME SDK 3.0, která podporuje vývoj CLDC a CDC aplikací. Další platformy se dají do tohoto IDE registrovat.

Například jeden z nastrojů, které umožní tvorbu jednoduchých aplikací i úplným začátečníkům je Visual Mobile Designer (VMD) umožňuje velmi rychlé vytvoření GUI, pomocí systému „drag and drop“ můžeme přidávat čekací obrazovky, přihlašovací obrazovky, prohlížeče souborů, editor SMS zpráv a další. Nástroj nazvaný Analyzer vyhledá nevyužité komponenty, které pak můžeme smazat a tím zmenšíme velikost výsledného souboru. Analyzer také zkontroluje, zda přidané komponenty jsou kompatibilní s MIDP 1.0. VMD jsem taky v počátcích zkoušel, ale opět jsem skončil u vlastního kódu.



Obr. č.2 : ukázka VMD

Na obrázku č.2 vidíme návrh jednoduché aplikace vytvořený ve VMD, obdélníky představují jednotlivé komponenty (MIDlet, textBox, formulář), do kterých můžeme přidávat příkazy jako je okCommand nebo exitCommand a poté tyto příkazy napojit na další moduly.

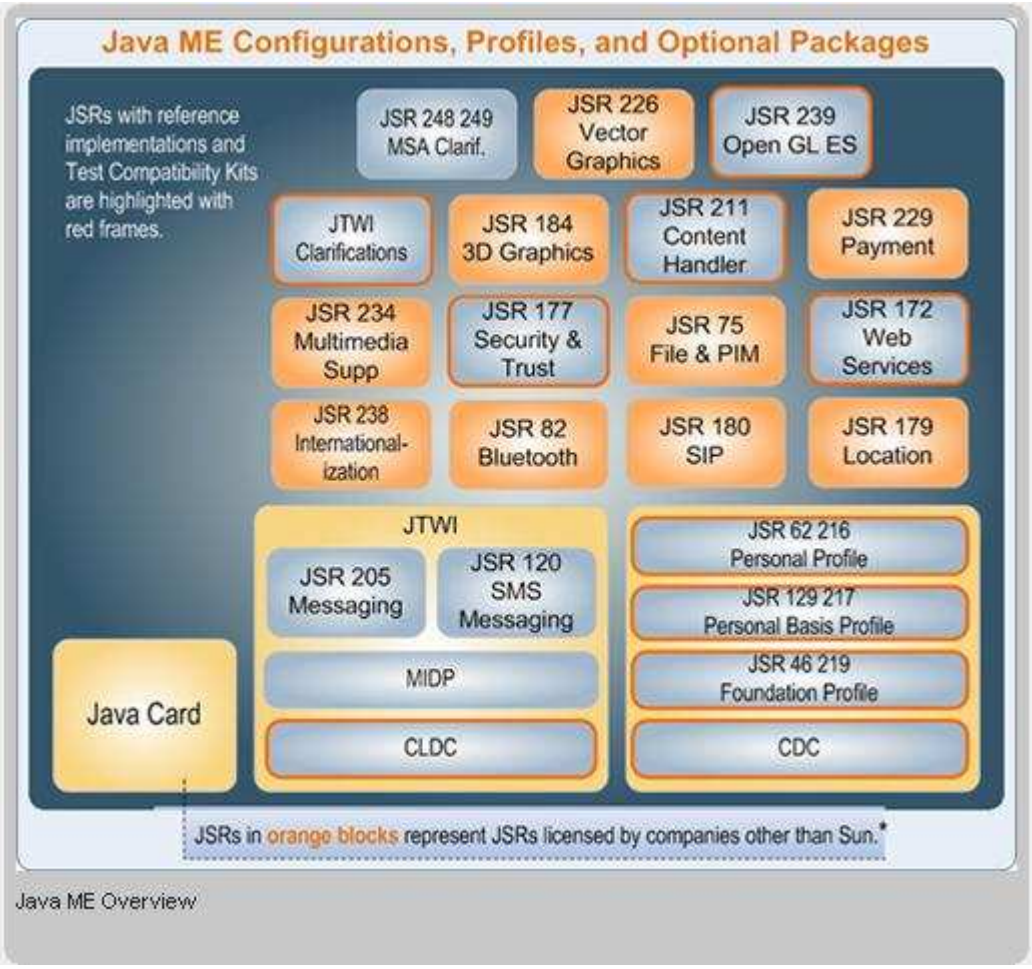
Můžeme navrhovat i vlastní komponenty a nebo SVG komponenty. Dále můžeme přidávat další konfigurace a s tím spojené emulátory pro testovaní na různých platformách například firem Nokia, Motorola a další. Lze zde napsat i aplikace s přístupem na webové služby.

2.2 Celkový pohled na Java ME

Na rozdíl od Java SE není Java ME jednolitým kusem software, aspoň ne v jeho pravé definici. Tento rozdíl může být matoucí obzvláště pro vývojáře, kteří jsou zvyklí na Java SE. Java ME je platforma, kolekce technologií a specifikací, které jsou navrženy pro odlišné části trhu s malými zařízeními. Protože má Java ME takové velké rozpětí, tak nemůžeme očekávat, že vytvoříme jedno řešení, které budeme moci aplikovat na všechna zařízení. Java ME je proto rozdělena do konfigurací, profilů a volitelných balíčků. Čerpáno z [6].

Konfigurace jsou specifikace, které upřesňují virtuální stroj (VM – Virtual Machine) a základní sadu API, které mohou být použity pro určitou třídu zařízení. Konfigurace může být například navržena pro zařízení, která mají méně než 512 KB paměti a občasně se připojují do sítě. Virtuální stroj je buď kompletní Java virtuální stroj (JVM – Java Virtual Machine) a nebo je nějakou jeho podmnožinou. Množina API je obvykle podmnožinou Java SE API.

Profil je postaven na konfiguraci, ale přidává více specifických API, aby vytvořil kompletní prostředí pro vytváření aplikací. Zatímco konfigurace popisovala Java virtuální stroj a základní množinu API, tak přesto nebyla natolik přesná, aby umožňovala vyvíjet kompletní aplikace. Profily obvykle zahrnují API pro životní cyklus aplikace, uživatelské rozhraní a pro trvalé uložení dat.



Obr. č.3 : Java ME - přehled

Volitelné balíčky poskytují další funkce, které nemusí být spojeny s určitou konfigurací nebo profilem. Příkladem volitelného balíčku je Bluetooth API (JSR – 82), které poskytuje standardizované API pro vytváření spojení pomocí Bluetooth. Tento volitelný balíček může být implementován s doslova jakoukoli kombinací konfigurací a profilů.

Jak ukazuje obrázek č.3 na předešlé straně, Java ME má dvě hlavní větve. První je založena na Connected Limited Device Configuration (CLDC). Tato konfigurace je pro malá bezdrátová zařízení jako je pager, mobilní telefon a PDA (Personal Digital Assistant). Profil MIDP, který je založen na CLDC, byl prvním dokončeným Java ME aplikačním prostředím. Zařízení s podporou MIDP jsou vysoce rozšířená.

Další hlavní větev Java ME je založena na Connected Device Configuration (CDC). Tato konfigurace je určena pro větší zařízení (z pohledu velikosti paměti a výpočetního výkonu procesoru) s masivní síťovou podporou. Set-top boxy a high-end PDA jsou dobrými příklady CDC zařízení.

Bezdrátová Java technologie je průnikem dvou rozsáhlých světů, bezdrátové datové komunikaci a Java platformy. Bezdrátová Java technologie zahrnuje tyto části: Java Card, Java ME, Java SE a Java EE.

Bezdrátová Java technologie a Java ME není to stejné. Na jedné straně, Java ME zahrnuje více než jen bezdrátová zařízení. Zatímco některé části Java ME jsou výhradně navržený pro bezdrátová zařízení, jiné části nejsou.

MIDP neznamená Java ME, MIDP je pouze první dokončený profil, který se stal se základem pro zařízení na celém světě, ale jak vidíme na obrázku, tak Java ME má plno dalších aspektů.

Platforma Java je bezpečná. Java kód se spouští v mezích Java virtuálního stroje, který poskytuje bezpečné prostředí pro spuštění kódu. Přenositelnost není úplně ideální pro Java bezdrátovou technologii. Jednu aplikaci sice můžeme spustit na více zařízeních. Například, když napíšeme MIDlet (MIDP aplikaci), tak ji můžeme spustit na každém zařízení, které implementuje specifikaci MIDP, ovšem tyto zařízení musí dodržovat specifikaci přesně podle standardu, jinak jsou jen problémy, jak jsem se přesvědčil i já při vývoji této hry. Více o této problematice naleznete v části implementace.

2.3 Bezdrátová technologie Bluetooth (JSR – 82)

2.3.1 Bluetooth API

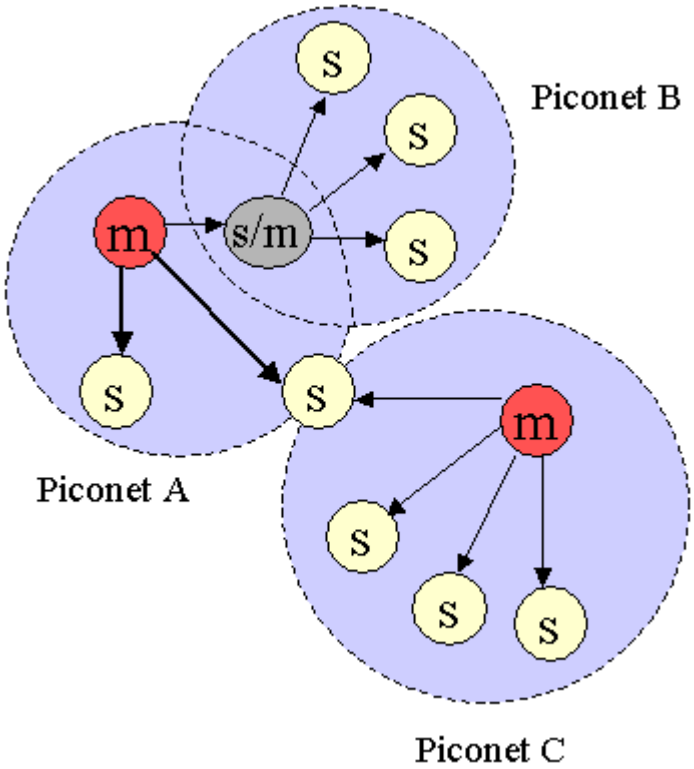
Toto API se skádá ze dvou balíčků – Bluetooth Core API (volitelné) a OBEX (Object Exchange) API. OBEX je nezávislé API sloužící k přenosu dat, které může být použito i bez Bluetooth Core API. Příslušné balíčky se nazývají javax.bluetooth a javax.obex. Tyto API neimplementují specifikace Bluetooth, pouze poskytují Bluetooth schopnosti pro zařízení s podporou Java ME. Čerpáno z [2] a [3].

2.3.2 Technologie Bluetooth

Jedná se o technologii rádiových vln, která využívá frekvenční pásmo ISM (Industrial-Scientific-Medical) 2,4GHz. Technologie Bluetooth byla oficiálně představena v roce 1994 švédským výrobcem mobilních telefonů firmou Ericsson jako cesta pro komunikaci mezi mobilními telefony na krátkou vzdálenost do 10 metrů.

V roce 1998 firmy jako Ericsson, IBM, Intel, Nokia a Toshiba založily asociaci Bluetooth Special Interest Group, aby zde vytvořily volně dostupnou specifikaci komunikace na krátkou vzdálenost. Od té doby se do této asociace přidalo dalších více než 2000 firem.

Výhody Bluetooth spočívají v tom, že je to bezdrátová technologie a zařízení se najdou navzájem automaticky bez zásahu uživatele (pokud není vyžadována autentizace), Bluetooth modul je levný, ISM pásmo není licencováno, podporuje datové i hlasové přenosy a to dokonce zároveň. Mezi dvěma zařízeními nemusí být přímá viditelnost, ale signál projde i přes zeď nebo přes materiál zavazadla ve kterém máte mobil.



Obr. č.4 : síťová topologie Bluetooth

Síťovou topologii Bluetooth můžeme vidět na obrázku č.4. Zařízení podporující Bluetooth jsou organizovány do skupin, které se nazývají „piconets“. Vysvětlení k obrázku č.4: **s** – slave; **m** – master; **s/m** znamená, že v jednom piconetu se chová jako master a ve druhém jako slave.

2.3.3 Profily Bluetooth

Profily Bluetooth se starají o součinnost mezi zařízeními a aplikacemi od různých výrobců a prodejců. Profil definuje role a schopnosti pro specifické typy použití. Bluetooth zařízení nemůže interagovat dokud se nepodřídí jednomu z profilů.

Generic Access Profile – definuje procedury spojení, rozpoznání zařízení a řízení propojení. Také definuje procedury vztažené k různým bezpečnostním modelům a požadavky na formát pro parametry přístupné na uživatelské úrovni. Tento profil je vyžadován jako nutné minimum, které musí zařízení podporovat.

Service Discovery Application and Profile – definuje vlastnosti a procedury, aby jedno Bluetooth zařízení mohlo rozpoznat služby nabízené jiným Bluetooth zařízením a získat potřebné informace vztahující se k té službě.

Serial Port Profile – definuje požadavky pro Bluetooth zařízení, která chtějí vytvořit propojení emulující sériový kabel s využitím RFCOMM protokolu.

LAN Access Profile – definuje jakým způsobem se Bluetooth zařízení může připojit k službám LAN pomocí PPP a ukazuje jak se pomocí PPP mechanismu dá vytvořit síť skládající se z Bluetooth zařízení.

Synchronization Profile – definuje požadavky na aplikace, které potřebují synchronizovat data mezi dvěma nebo více zařízeními.

2.3.4 Zabezpečení Bluetooth

Bezpečnost je poskytována ve 3 možnostech: pseudo náhodné měnění signálu, autentizace a šifrování.

Menění frekvence – díky tomuto opatření je složitější odposlouchávání signálu

Autentizace – umožňuje uživatelům limitaci spojení pro určitá zařízení

Šifrování – využívá šifrovacích klíčů k tomu, aby data byla srozumitelná jen pro autorizované skupiny

Všechny Bluetooth zařízení musí implementovat Generic Access Profile, který obsahuje všechny protokoly Bluetooth a možná zařízení. Tento profil také definuje bezpečnostní vzor, který obsahuje tři bezpečnostní módy:

1. **Mode 1** není bezpečný mód, protože žádné bezpečnostní procedury nejsou spuštěny.

2. **Mode 2** je známý jako vynucená bezpečnost na servisní úrovni. Pokud zařízení pracují v tomto módu, tak žádné bezpečnostní procedury nejsou spuštěny dokud se nevytvoří spojení. Tento mód umožňuje aplikacím, aby měly různé politiky přístupu a spouští je paralelně.

3. **Mode 3** je známý jako vynucená bezpečnost na úrovni spojení. V tomto módu jsou bezpečnostní procedury spuštěny ještě před tím, než je navázáno spojení.

2.3.5 Příklady Bluetooth tříd

Třída DiscoveryAgent

Tato třída má na starost rozpoznání zařízení a jeho služeb, které nabízí. Klientská aplikace, která čeká až bude upozorněna na to, že dostupná zařízení a služby byly rozpoznány musí implementovat a zaregistrovat rozhraní DiscoveryListener, které definuje odpovědi na zprávy o rozpoznání zařízení a služeb.

Třída UUID

V Bluetooth je každá služba a každý atribut služby unikátně definován pomocí Universally Unique Identifier (UUID). Jak naznačuje překlad zkratky UUID Univerzální Unikátní Identifikátor, tak každý identifikátor má zajištěno, že je unikátní v prostoru a čase. Třída UUID reprezentuje krátké (16 nebo 32 bitové) a dlouhé (128 bitové) UUID. Poskytuje konstruktory pro vytvoření UUID z hodnoty typu String a nebo z hodnoty o velikosti 16 nebo 32 bitů, dále metodu pro porovnání dvou UUID (jestliže jsou oba hodnoty 128 bitů) a metodu na konvertování UUID do hodnoty String. Instance UUID je neměnná a pouze služby identifikovatelné pomocí UUID jsou rozpoznatelné.

Schopnosti JSR – 82:

- řídit nastavení lokálního Bluetooth zařízení
- nalézt jiná zařízení v okolí dosahu signálu
- vyhledat dostupné služby, které tato zařízení nabízí
- připojit se na tyto služby a komunikovat s nimi
- zaregistrovat službu na lokální Bluetooth zařízení a umožnit dalším zařízením s ní komunikovat
- řídit a kontrolovat komunikaci
- poskytovat ochranu pro výše zmíněné možnosti

3. MIDP a Game API

3.1 MIDP 1.0

První verze MIDP 1.0 postrádala lepší grafické možnosti, což vyústilo v omezenou podporu pro tvorbu her. Někteří prodejci zařízení se snažili tyto limitace vyřešit přidáváním vlastních tříd, ale jejich používání vedlo ke špatné přenositelnosti aplikace na jiná zařízení. MIDP 2.0 se těchto limitací zbavila tím, že definovala nový balíček Game API, javax.microedition.lcdui.game. MIDP 1.0 postrádalo podporu pro výpočty s pohyblivou řadovou čárkou, díky tomu bylo složité 3D renderování, nedisponovalo podporou pro audio, pouze umožňovalo přehrát jednoduché systémové zvuky, aplikace nemohli zapisovat nebo číst z/do jednotlivých pixelů a díky tomu nebylo možno provádět základní manipulace s obrázky, jediný podporovaný síťový protokol byl http.

MIDP 1.0 poskytuje GUI jak na vysoké, tak i na nízké úrovni díky třídám Screen a Canvas. Typická herní aplikace například začíná obrazovkou s vyobrazením hlavního menu a dále se tam uplatní další GUI komponenty jako forms, alert, textbox, list. Canvas je však tou třídou, kde se odehrávají všechny grafické operace. Na obrázku č.5 jsem přiložil typickou ukázku kostry hry, která používá Canvas.

```
public class MyCanvas extends Canvas implements Runnable {

   public void run() {
      while(true) {
         repaint(); // update the game state
   }

   public void paint(Graphics g) {
      // code for painting
   }

   protected void keyPressed(int keyCode) {
      // respond to key events
   }

}
```

Obr. č.5 : ukázka Canvas

Problémem u tohoto návrhu je, že herní logika je rozeseta do tří rozdílných metod, takže vše běží v různém vlákně a při běhu aplikace se můžeme dočkat problémů jak s vykreslováním grafiky, tak s odezvou na stisk kláves. Tyto problémy byli vyřešeny příchodem MIDP 2.0 a balíčkem Game API.

3.2 Game API

3.2.1 Balíček Game API

javax.microedition.lcdui.game poskytuje třídy, které umožňují herním vývojářům vytvářet hry s bohatým grafickým obsahem. API je založeno na nízko úrovňové grafice s třídami jako Graphics a Image, které jsou známy z MIDP 1.0 a zároveň přidává vysoko úrovňovou grafiku díky balíčku Game API. Čerpáno z [4], [5] a [7].

Game API obsahuje 5 tříd:

GameCanvas je podtřídou javax.microedition.lcdui. Canvas a poskytuje základní herní funkce pro aplikaci. Obsahuje herně orientované metody jako zjišťování stavu herních kláves a synchronní vykreslování grafiky. Informaci o stisku kláves můžeme získat zavoláním metody getKeyStates(), která dokonce registruje i stisk několika kláves najednou. Další výhodou je, že logika hry neběží v několika vláknech, ale jen v jednom jak vidíme na obrázku č.6. Typický herní cyklus se skládá z kontroly stisku kláves, herní logiky, a obnovováním grafiky a uživatelského rozhraní.

```
public class MyCanvas extends GameCanvas implements Runnable {

   public void run() {
      Graphics g = getGraphics();
      while(true) {
         // update the game state
         // ...
         int k = getKeyStates();
         // respond to key events
         flushGraphics();
      }
   }
}
```

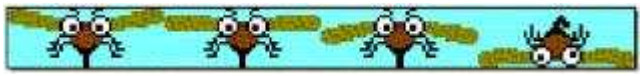
Obr. č.6 : ukázka GameCanvas

Layer je abstraktní třída, která poskytuje základní kostru pro grafiku ve vrstvách. Reprezentuje ve hře vizuální elementy jako je Sprite a TiledLayer s atributy jako je poloha, velikost (šířka, výška) a viditelnost.

Sprite je základní animovaná vrstva a může zobrazovat několik grafických snímků, které jsou stejné velikosti a uloženy v jednom objektu typu Image. Sprite podporuje transformace jako převrácení a rotaci a dále detekci kolizí. V typické dvourozměrné hře se nachází herní postavičky, které se pohybují a případně se dostávají do vzájemné interakce, v Game API jsou nazývány Sprite. Třída Sprite reprezentuje obrázek v jistém bodě času. Můžeme definovat sekvenci snímků a jejich pořadí v jakém se mají objevovat. Na tyto snímky můžeme aplikovat jednoduché transformace jako je rotace a nebo zrcadlení. Sekvence snímků se dají uložit do jediného obrázku a metody třídy Sprite si ten obrázek sami rozparcelují na jednotlivé snímky, když zadáme šířku a výšku jednoho snímku.

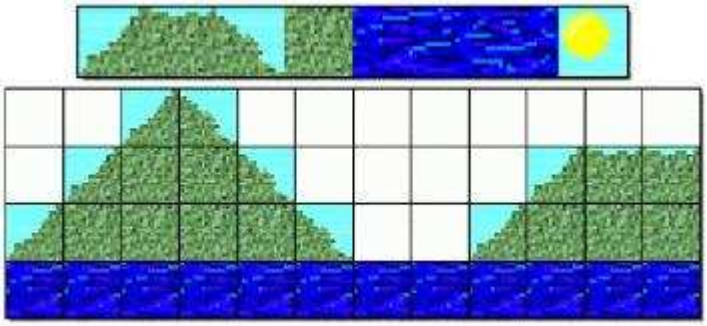
Na přiloženém obrázku č.7 lze vidět jak můžeme mít takovou animaci o čtyřech krocích uloženou. Každému snímku je přiřazeno unikátní číslo od 0 až dále, podle počtu snímků, tato sekvence je uložena v poli celých čísel, a tím je určeno pořadí vykreslení jednotlivých snímků. Můžeme si určit i jinou než standardní sekvenci a to tak, že se můžeme přeskakovat z jednoho snímku na jiný a nebo některý snímek opakovat vícekrát za sebou. Třída Sprite dále implementuje koncepci referenčního pixelu, kdy si v samotném spritu stanovím tento referenční bod a transformační operace se pak budou provádět ne podle bodu v levém horním rohu spritu, ale podle nového bodu, který jsme nastavili. Různé transformace mohou být aplikovány na sprity. Rotace o 90° (180°, 270°) a zrcadlení podle vertikální osy. Sprite může být kdykoli vykreslen zavoláním metody paint(Graphics).

Můžeme dále definovat obdélník pro detekci kolizí, tento obdelník může být stejně velký jako Sprite, nebo může být větší a nebo menší. U Spritu můžeme detekovat kolize s objektem typu Sprite, TiledLayer a Image. Kolize se detekují dvěma způsoby a to na úrovni jednotlivých pixelů, což je náročnější a přesnější metoda a nebo na základě výše zmíněného obdélníka pro detekci kolizí.



Obr. č.7 : ukázka Sprite animace

TiledLayer reprezentuje mřížku buněk, které mohou být vyplněny dlaždicovými obrázky (ty mohou být i animované). Tato třída umožňuje tvůrcům vytvořit rozsáhlé grafické oblasti bez toho, aby zvyšovali nároky na systémové prostředky, jak by tomu bylo v případě použití jednoho velkého obrázku. Zdrojový obrázek se podobně jako v případě animovaného spritu skládá z různého počtu menších stejně velkých obrázků, které mají své číslo a my pak v matici určíme, který obrázek se kde vykreslí a tím vytvoříme herní plochu s menším zatížením systému než jak by tomu bylo v případě použití jednoho velkého obrázku. Dlaždice se načítají z obrázku, kde má každá dlaždice stejný rozměr Width (šířka) a Height (výška) z toho plyne, že šířka obrázku ze kterého se dlaždice čtou je celočíselným násobkem šířky dlaždice a výška obrázku je celočíselným násobkem výšky dlaždice. Tento princip je zachycen na obrázku č.8, kde z vrchního zdrojového obrázku vytvoříme obrázek spodní.



Obr. č.8 : využití TiledLayer

LayerManager zjednodušuje vývoj her díky automatizaci vykreslovacího procesu. Tato třída je užitečná pro hry s více vrstvami. Umožňuje vývojářům určit oblast, která reprezentuje herní okno, které uvidí hráč a automaticky vykresluje všechny vrstvy do tohoto pohledu ve správném pořadí. Umožňuje řídit a organizovat grafické vrstvy ve hrách. Například uděláme pozadí s použitím třídy TiledLayer a hráčskou postavu ze třídy Sprite, obě tyto třídy jsou podtřídami třídy Layer. Pořadí v jakém jsou vrstvy vykreslovány je opak pořadí jejich přidání do LayerManageru, jinými slovy, první vrstva, kterou přidáme příkazem append do LayerManageru bude vykreslena poslední. Další důležitá funkce umožní vykreslit grafickou oblast větší než je displej zařízení a poté vybrat, která část a kdy se bude na displeji zobrazovat. Layer manager je série vrstev. Vrstvy mohou být do LayerManageru přdány (append), odebrány (removed) a vloženy (inserted) mezi již existující vrstvy. Jestliže je vrstva vyjmuta, indexy se přepočítají, tak aby bylo dodrženo správné pořadí vykreslovaní. Výhodou LayerManageru je metoda setViewWindow(int x, int y, int width, int height), kde určíme oblast, kterou uživatel po spuštění uvidí a pokud je herní plocha větší než ta ohraničená, tak voláním metody setViewWindow a úpravou souřadnic x a y můžeme například herní plochu posouvat doprava, pokud uživatel stiskne klávesu pro pohyb vpravo, tím získáme efekt, že se například postava na displeji bude pohybovat směrem doprava.

MIDP 2.0 s balíčkem javax.microedition.lcdui.game je sytém, který usnadňuje vývoj her pro zařízení s podporou MIDP. Game API balíček poskytuje 5 velmi dobrých tříd, které umožňují vytvářet bohatý herní obsah pro mobilní telefony. Třídy byly implementovány, aby významně ulehčily vývoj, zredukovaly výslednou velikost aplikací a zlepšily výkon.

4. Implementace

4.1 Úvod

Pro vývoj aplikace jsem si vybral vývojové prostředí NetBeans 6.5. Jako platformu pro emulaci mobilního telefonu jsem zvolil emulátor dodávaný přímo firmou Sun a to Sun Java(TM) Wireless Toolkit 2.5.2 for CLDC. Zvolil jsem tento přístup hned z několik důvodů:

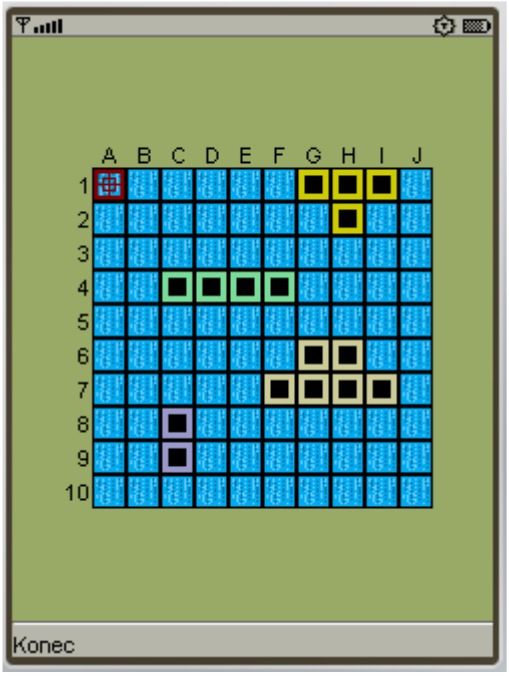
1) Jsem toho názoru, že bakalářská práce by měla být dělána podle standardů a ne pro mobily jen určité série a jen od jedné firmy, protože každá z těchto firem si tam dává nějaké svoje věci a nebo mění ty standardní. Samozřejmě pokud by šlo o komerční nasazení mé aplikace a chtěl bych, aby si moji hru mohlo zahrát co nejvíce lidí, tak bych samozřejmě musel udělat třeba 10 verzí své hry, aby pokryla co největší množství prodávaných telefonů, což nebylo náplní této práce.

2) Neměl jsem k dispozici mobilní telefon, který by podporoval technologii na kterých je moje aplikace postavena.

3) Na počátku vývoje jsem něco málo jsem zkoušel s emulátory firmy Nokia, ale setkal jsem se s velkými problém co se týká jen samotného spuštění emulátoru a to nemluvím o různých nestabilních stavech emulátoru, i když jsem na něm spustil jen obyčejný HelloWorld program. Proto jsem zvolil cestu bezproblémového vývoje aplikace pro vestavěný emulátor od firmy Sun, která dodržuje všechny standardy

Když jsem se začal seznamovat s tvorbou aplikací pro mobilní telefony a s tvorbou herního obsahu, tak jsem došel k závěru, že svou aplikaci postavím na technologii MIDP 2.0 a na Game API, které přináší 5 hlavních tříd pro tvorbu herního obsahu. Nejdříve jsem se začal seznamovat s tím, jak udělat hlavní obrazovku, kde by byla položka Menu, ve které by se dali vybírat jednotlivé možnosti. Jakmile se mi to podařilo, začal jsem experimentovat s třídou GameCanvas, kde jsem se seznámil jak kreslit základní útvary jako je čára, čtverec atd. Poté jsem se naučil pracovat s třídami jako je Sprite, TiledLayer a přidávat tyto prvky do LayerManageru.

4.2 Popis hry



Obr. č.9 : ukázka hry

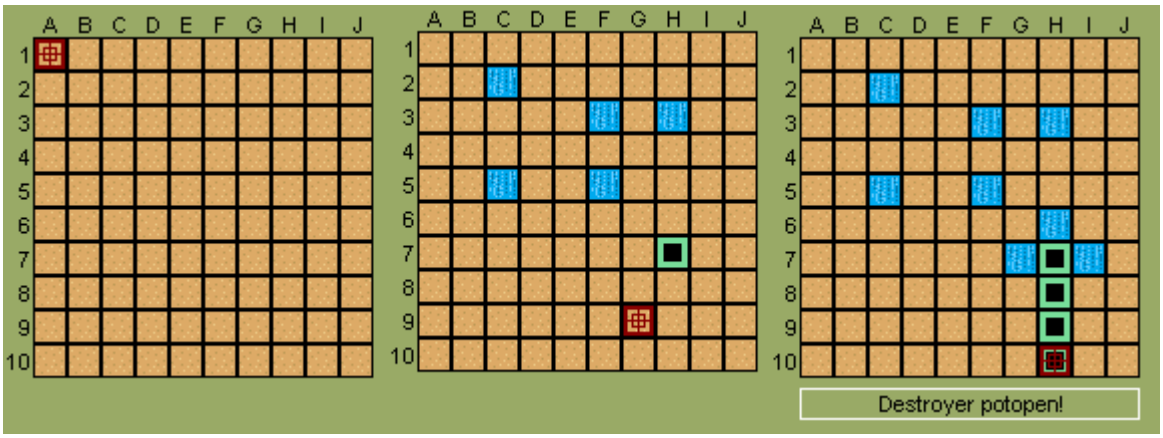
Na obrázku č.9, kde jsem pro tyto účely odkryl překrytí vidíme 4 typy lodí a to: Destroyer skládající se ze 4 objektů typu Sprite, Minesweeper skládající se ze 2 objektů typu Sprite, Cruiser skládající se ze 4 objektů typu Sprite a nakonec AircraftCarrier skládající se ze 6 objektů typu Sprite. Pro každou z těchto lodí je vyhrazena jedna třída. V této třídě sestavíme jednotlivé typy lodí, tak aby byly umístěny podle zadaných parametrů. Parametry jsou osa x a osa y, které určují pozici levého horního rohu prvního spritu a podle jeho pozice se pak k němu připojují další sprity. Další parametr je natočení lodě, kde číslo 0 znamená normální pozici, číslo 1 natočení o 90°, číslo 2 natočení o 180° a nakonec číslo 3 je natočení o 270°. Posledním parametrem je typ lodi. Rozmístění lodí na herní ploše se provádí náhodně, je to z toho důvodu, že jsem nechtěl zdržovat začátek hry, kdy by si hráč musel umisťovat jednotlivé lodě sám, jak jsem odpozoroval sám na sobě, tak když jsem si manuálně v kódu zkušebně umisťoval lodě, tak jsem tuto proceduru chtěl mít co nejdříve za sebou a volil jsem jednoduché rozmístění lodí. Navíc, když se lodě generují náhodně, tak je to i větší napětí ze hry, protože sám hráč neví, kde jsou jeho lodě umístěny.

Ačkoli náhodné generování lodí vypadalo z počátku jako jednoduchá záležitost nebylo tomu tak, po prvním pokusu jsem na obrazovce dostal chaos, kdy lodě byli vykresleny přes sebe a i mimo hrací plochu a tím pádem špatně. Proto jsem to vylepšil metodou JdeUmistit(x, y), která na souřadnice x a y volá metodu zásah a díky tomu zjistím, zda se vygenerovaný sprite dá umístit nebo ne. Protože pokud metoda zásah vrátí hodnotu true, tak metoda JdeUmistit vrátí false a tím se celý proces umisťování lodě zastaví a vygenerují se nové souřadnice pro umístění prvního spritu. Generování trvá do té doby, než jsou všechny sprity umístěny a tím je proces umisťování lodí dokončen. Jen bych podotknul, že čísla umístění se generují pro každý typ lodí podle jiných pravidel, protože loď, která se skládá ze dvou spritů bude mít více možností umístění než loď skládající se ze šesti spritů.

Po otestování jsem přišel na další nepříjemnost a to, že se lodě umístili tak, že se dotýkali hranou spritu a to nevypadalo vůbec dobře, někdy se vygenerovalo takové rozestavění, ve kterém všechny 4 lodě tvořili jeden celek. Proto jsem přistoupil ke zmapování těchto kritických oblastí pro každý typ lodě a každé natočení a tyto zjištěné oblasti, kterých bylo více než 200 jsem zahrnul do metody JdeUmistit a díky tomu se lodě rozmístí podle mých požadavků, to znamená že se vůbec nedotýkají. Konečné rozmístění každé lodi se připojí do LayerManageru a poté se vykreslí.

Dále je pro loď důležitá metoda trefMe(x, y), kde poté co je loď zasažena, neboli byl identifikován zásah na jeden ze spritů lodě, tak tato metoda přehraje zvuk výbuchu, musel jsem zajistit, aby se zvuk přehrál pouze jedenkrát, protože původně se přehrával i tehdy, když jsem omylem nebo úmyslně stisknul klávesu pro výstřel na již jednou odkrytém poli.

Další funkce této metody je indikace potopení celé lodi, kdy se přehraje jiný zvuk indikující potopení. To je zajištěno tím, že mám uloženy pozice spritů a po zasažení spritu změním pro něj vyhrazenou proměnnou z 0 na 1, pokud je u každého spritu proměnná nastavena na 1, to znamená, že byli zasaženy všechny sprity, ze kterých se loď skládala, tak se přehraje zvuk pro potopení a zároveň se na určenou dobu vypíše na obrazovku text, který nás informuje o druhu potopené lodi. Podobný text se objeví při zničení dalších typů lodí a nakonec se objeví informace o konci hry, pokud jsou všechny 4 lodě potopeny.



Obr. č.10 : průběh hry

Na obrázku č.10 vidíme zachycené tři obrazovky z různého průběhu hry. První obrazovka zachycuje hru na začátku, kdy jsou všechny pole zakrytá a v levém horním rohu je umístěn zaměřovací kříž. Navíc jsem přidal horizontální a vertikální pojmenování jednotlivých řad a sloupců a to z toho důvodu, že by například moji hru pozoroval i někdo jiný a chtěl by mi poradit další tah, tak stačí dlaždici k odkrytí jednoznačně identifikovat kombinací písmene a čísla. Poté se můžeme začít pohybovat zaměřovacím křížem do 4 směrů (kromě situace, kdy se zaměřovač dotýká okraje). Jakmile vybereme dlaždici, kterou chceme odkrýt stiskneme potvrzovací střed na emulátoru a odkryjeme dlaždici nad kterou byl zaměřovací kříž. Pak uvidíme buď moře (vedle) a nebo loď (zásah).

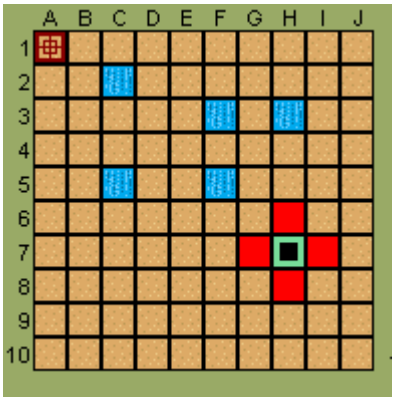
Pohyb zaměřovacího kříže jsem musel upravit a to tak, že původně stačilo jen lehce stisknout třeba směr doprava a kříž se pohnul i o pět dlaždic daným směrem. Tento problém jsem vyřešil uspáním vlákna na dobu 200 milisekund po každém stisku klávesy, tím jsem docílil požadovaného pohybu zaměřovače.

Hra dvou hráčů je vyřešena jednoduše a to tak, že udělám novou herní plochu a vygeneruji nové lodě a pak se pomocí kontextové klávesy hráči přepínají mezi první a druhou herní plochou, kdy vlastně hráč, který začíná má hned zobrazenou plochu soupeře a střílí přímo na ni a po jeho výstřelu jeden z hráčů zmáčkne pravou kontextovou klávesu a dostane se tak na plochu soupeře. Takhle se hráči přepínají dokud jeden z nich nepotopí všechny soupeřovy lodě a nestane se vítězem.

Možnost 1 hráč trénink je ideální pro rychlou hru, kdy se hráč snaží potopit všechny lodě a nehraje proti počítači, ani proti jinému hráči, tento typ hry je nejkratší a dají se díky němu dobře natrénovat barvy a tvary lodí pro ostrou hru.

Hra proti počítači mě stála nemalé úsilí při vymyšlení toho jak se má počítač chovat. Musím se přiznat, že můj původní záměr, aby se počítač choval jako člověk nebyl naplněn. Hodně jsem musel experimentovat, než jsem vymyslel aspoň trochu slušnou počítačovou inteligenci. Na jedné straně bych lehce mohl udělat dokonalou inteligenci, která by každým odkrytím dlaždice trefila cíl, ale toto by nikoho nebavilo a na druhé straně stojí „inteligence“, kde by docházelo pouze k náhodnému odkrývání dlaždic, bez jakékoli reakce na případný zásah. V mém případě jsem inteligenci realizoval následujícím způsobem: počítač začíná náhodným výstřelem, pokud nic nezasáhne, tak se vygeneruje další náhodná dlaždice, která se odkryje, to se děje tak dlouho dokud nezasáhne loď, poté se začne postupně prohledávat okolí této dlaždice a to ve čtyřech směrech, jak je zachyceno na obrázku č.11 červenou barvou. Pokud prohledá tyto 4 směry (vynechá ty které už byly odkryté) a neproběhne zásah, tak se vygeneruje další náhodné číslo a jde hledat jinde, pokud proběhne zásah, tak zbývající směry už neodkrývá a věnuje se prohledávání okolí dalšího zásahu.

Tato metoda je nedokonalá v tom, že pokud proběhne zásah uvnitř lodi, tak se počítač vydá prohledávat jen na jednu stranu lodi a jakmile dojede na konec lodi a neproběhne další zásah, tak skočí na jinou náhodně vygenerovanou dlaždici, i když by mu k potopení lodi chybělo zasáhnout poslední sprite, ale ten byl na druhé straně než na kterou prohledával okolí, takže tento sprite zůstal nezasažen a k jeho zásahu a případnému potopení lodi dojde až někdy v dalších kolech, kdy bude vygenerováno číslo příslušné dlaždice. Jak lze vidět, tak můj systém má své chyby, ale myslím si, že jistá známka inteligence tam je.



Obr. č.11 : možné dlaždice pro odkrytí

Hra přes Bluetooth je realizována odlišně od hry dvou hráčů na jednom zařízení. Nejdříve se na jednom zařízení spustí server, který čeká na připojení klienta. Pak se na druhém zařízení spustí klient a ten se připojí na server, poté se hned zobrazí hrací plocha. Která je na obou zařízeních stejná, to znamená na každém zařízení jsou stejně rozmístěné lodě, pro demonstrační účely jen v jedné konfiguraci rozmístění. Poté začíná hru hráč na zařízení, které je server, klient má v tu dobu zablokované ovládání, hráč server určí pole k odkrytí a stiskne potvrzovací střed. Poté se odkryje dlaždice, zablokuje se ovládání na straně serveru a číslo odkryté dlaždice se přenese na klienta, kde se tato dlaždice také odkryje. Nyní následuje hra na straně klienta, ten odkryje nějakou další dlaždici k odkrytí a poté se opět zablokuje ovládání na straně klienta a číslo této dlaždice se pošle na stranu serveru, který u sebe také odkryje tuto dlaždici. Hra tedy probíhá na jedné hrací ploše, kdy se v odkrývání dlaždic střídají oba hráči a vyhrává ten, který jako poslední potopí závěrečnou loď.

5. Závěr

Náplní této práce byla implementace hry Lodě pro mobilní telefon na platformě Java ME. Zdokonalil jsem se v programování v jazyce Java a hlavně jsem se seznámil s vývojem aplikací pro mobilní telefony, zdokonalil jsem se taktéž v práci s vývojovým prostředím NetBeans (v mém případě byla použita verze 6.5 s integrovaným Mobility packem).

Zadání práce jsem splnil, ovšem jako všechno co člověk dělá i toto se dalo udělat lépe, proto mám svůj osobní cíl se touto prací zabývat i po jejím odevzdání a hru Lodě řádně vylepšit hlavně co se týká umělé inteligence a hře pomocí Bluetooth, abych ji pak případně mohl uvést jako volně šiřitelnou aplikaci zdarma ke stažení na svém budoucím webu.

Použitá literatura a materiály:

[1] NetBeans [online] <http://www.netbeans.org>

[2] JSR-82 : Java Bluetooth [online] <http://www.jsr82.com/jsr-82-basics/>

[3] Wireless Application Programming with Java ME and Bluetooth [online]
<http://developers.sun.com/mobility/midp/articles/bluetooth1/>

[4] Getting Started With the MIDP 2.0 Game API [online]
<http://developers.sun.com/mobility/midp/articles/gameapi/>

[5] Overview (MID Profile) [online] <http://java.sun.com/javame/reference/apis/jsr118/index.html>

[6] Introduction to Mobility Java Technology [online] <http://developers.sun.com/mobility/getstart/>

[7] Creating 2D Action Games with the Game API [online]
<http://developers.sun.com/mobility/midp/articles/game/>

Seznam příloh na CD

1. hra Lodě jako projekt do NetBeans
2. programátorská příručka jako vygenerovaný JavaDoc
3. uživatelský manuál